УДК 651.326

В.Д. ДАЛЕКА, НТУ "ХПИ",
С.С. ВДОВИЧЕНКО, НТУ "ХПИ"

ИССЛЕДОВАНИЕ РАСПРЕДЕЛЕНИЯ ПАМЯТИ КОМПЬЮТЕРА ПРИ ОБЪЕКТНО-ОРИЕНТИРОВАННОМ ПРОГРАММИРОВАНИИ

Розглянуто питання виділення, розподілення та використання фізичної (оперативної) пам'яті, що виділяється для екземплярів класу (об'єктів), у IBM PC сумісних комп'ютерах при реалізації основних ідей, принципів та концепцій об'єктно-орієнтованого програмування.

The questions of excretion, allocation and using of core quota for examples of class (object) on IBM PC conformant computers for realization of main ideas, principles and conceptions of object-oriented programming are studied.

Постановка проблемы. Вопросы распределения памяти компьютера для данных, объектов и структур различных форматов, а также поиска оптимальных алгоритмов их обработки были и остаются одними из наиболее важных в области высоких технологий, при компьютерной обработке данных. Знания принципов и механизмов распределения памяти необходимы при создании прикладных и системных программных продуктов, например, при разработке 2D и 3D-графики, СУБД, сетевых и Web-приложений, систем безопасности и ограничения доступа и т.п. Суть работы каждого программиста заключается в создании структур данных и разработке алгоритмов их обслуживания. И от того, насколько информирован программист о том как, когда и сколько памяти выделяется для переменных, зависит эффективность его работы и показатели качества его программного продукта в целом.

Анализ литературы. Успех и популярность любой структуры данных зависит от удобства и эффективности ее применения, что, в свою очередь, зависит от удачного проектирования самой структуры и алгоритмов ее обработки. Эти вопросы, особенно вопросы эффективности, очень тесно связаны с порядком выделения, распределения и использования оперативной памяти для данных той или иной структуры и программного кода, реализующего алгоритмы их обработки. В литературе [1, 2] достаточно подробно рассмотрены вопросы выделения и распределения памяти для данных базовых типов, статических и динамических структур, но при этом практически не рассматриваются такие типы данных как классы. Имеются, правда, электронные издания [3 – 6], в которых приведены фрагментарные данные по интересующему вопросу. Литература, посвященная аппаратным средствам IBM PC [8] описывает структуру самой оперативной памяти на физическом и логическом уровне без описания размещения в ней каких-либо структур данных.

Цель статьи. Исследовать выделение, распределение и использование

оперативной памяти IBM PC совместимых компьютеров при реализации основных идей, принципов и концепций объектно-ориентированного программирования (ООП). Рассмотреть распределение оперативной памяти для объектов простых классов, классов-наследников при многоуровневом и множественном наследовании. При этом основное внимание акцентировать на размещении в памяти статических и нестатических данных-членов (свойств) классов как ключевых структурных элементов. Такие же технологии и концепции как конструкторы и деструкторы, полиморфизм, виртуальное наследование в данной статье не рассматриваются.

Если вспомнить, что структуры данных и их оптимизация – это один из важнейших вопросов при разработке практически любого программного продукта, то актуальность и своевременность темы, затронутой в статье, становиться очевидной.

Выделение и распределение оперативной памяти компьютера для объектов при ООП. Объектно-ориентированное программирование не является последним и самым современным стилем программирования, но этот стиль или технология является основной и базовой идеологией практически для всех современных технологий и стилей, применяющихся в мире программирования. Вышесказанное дает основание утверждать, что идеи и принципы ООП являются в данный момент наиболее распространенными в программировании, а, следовательно, знания принципов выделения и распределения памяти в ООП окажутся полезными как начинающим, так и опытным программистам.

Для проведения исследований был выбран язык программирования C++ как один из самых популярных языков, наиболее полно поддерживающий идеи, принципы и парадигмы ООП, сочетающий в себе средства осмысленной связи данных и алгоритмов, что дает возможность расширять и структурировать программы. Кроме того, что очень важно в данной работе, C++ обладает необычайной гибкостью при работе с памятью.

Как показывает практика, вопросы выделения памяти чрезвычайно чувствительны к изменению программного и аппаратного обеспечения, на котором они рассматриваются. Поэтому детали и тонкости реализации некоторых аспектов могут претерпевать значительных изменений не только при переходе от одной аппаратной платформы к другой, но и при смене операционных систем и средств разработки от разных производителей и даже при переходе от версии к версии компиляторов одного производителя. Следует отметить, что приводимые ниже данные получены на компьютере класса IBM PC, работающим под управлением ОС Microsoft Windows XP Professional SP2, в среде разработки Microsoft Visual Studio 98. Экспериментально проверялись объем и порядок выделения стековой и динамической памяти для данных-членов классов, ее распределение для объектов классов и были получены такие результаты.

1. Объекты простых классов, не содержащие данные и функции (***"пустые» объекты"***), как оказалось, требуют 1 байт памяти. Несколько объектов одного или нескольких подобных классов размещаются и в стековой, и в динамической памяти в порядке убывания адресов ячеек памяти (табл. 1).

Таблица 1

Распределение памяти для объектов "пустых" классов

Стековая память			Динамическая память		
Имя	Адрес	Размер (байт)	Имя	Адрес	Размер (байт)
objClass1	0x0012FF7C	1	objClass1	0x00491EA0	1
objClass2	0x0012FF78	1	objClass2	0x00491E70	1
objClass3	0x0012FF74	1	objClass3	0x00491E50	1

На порядок расположения объектов не влияет порядок объявления их классов и то, в какой (локальной или глобальной) части программы они были объявлены. Объекты располагаются в том порядке, в котором они определенны в программе. Объекты, размещенные в стековой памяти, располагаются на расстоянии 4 байта друг от друга, что является следствием оптимизации кода, активно используемой современными компиляторами.

2. Объекты простых классов, содержащие данные-члены базовых и интегрированных типов, вложенные объекты, указатели, как и предполагалось, требуют память, размер которой равен суммарному размеру всех его данных-членов, независимо от того, являются ли они открытыми, закрытыми или защищенными. Данные-члены располагаются в памяти, начиная с первого байта выделенной памяти в направлении возрастания адресов ячеек стековой или динамической памяти в том же порядке, в каком они описаны в объявлении класса (табл. 2).

Таблица 2

Распределение памяти для объектов простых классов

Данные классов		Объект класса cOwner		Дамп памяти	
cMember	cOwner	Адрес	Размер	Адрес	Контент

public:
int int1 = 4;
protected:
int int2 = 5;
private:
int int3 = 6;
	public:
	int int1 = 1;
	protected:
	int int2 = 2;
	cMember member;
	private: int int3 = 3;
		0012FF68	24 байта	0012FF68
				0012FF6C
				0012FF70
				0012FF74
				0012FF78
				0012FF7C
					00000001
					00000002
					00000004
					00000005
					00000006
					00000003

Таким образом, размер не наследуемого и не содержащего виртуальных функций объекта в общем случае равен суммарному размеру всех его данных-членов. При оптимизации кода компилятор может выделять для таких объектов дополнительную память.

3. Статические данные-члены **(**открытые, закрытые и защищенные) располагаются в отдельном сегменте памяти, как было отмечено в [7]. Они

принадлежат классу, но не принадлежат его экземпляру. Статические данные-члены размещаются в выделенной для них памяти подряд, в порядке увеличения адресов ячеек памяти (табл. 3).

Таблица 3

Распределение памяти для объектов простых классов со статическими данными

Данные классов			Дамп памяти для статических данных	
cMember	cOwner	Статические	Адрес	Контент

public:
static int statm1;
int mint1=1;
protected:
static int statm2;
int mint2=2;
private:
static int statm3;
int mint3=3;
	public:
	static int stato1;
	int oint1 = 7;
	protected:
	static int stato2;
	int oint2 = 8;
	private:
	static int stato3;
	int int3 = 9;
	cMember member;
		int cMember::statm1=4;
		int cMember::statm3=6;
		int cOwner::stato1=10;
		int cOwner::stato3=11;
		int cMember::statm2=5;
		int cOwner::stato2=12;
			0047DDD0
			0047DDD4
			0047DDD8
			0047DDDC
			0047DDE0
			0047DDE4
				00000004
				00000006
				0000000a
				0000000b
				00000005
				0000000c

При этом последовательность расположения данных определяется исключительно порядком их определения и инициализации за пределами объявления классов, и не зависит ни от порядка объявления самих классов, ни от порядка создания их объектов. Размер памяти для объекта класса cOwner составляет 24 байта.

4. Простое наследование (как открытое, так и закрытое) приводит к копированию всех открытых, закрытых и защищенных данных базового класса в производный класс. Данные базового класса размещаются вначале области памяти, выделенной для объекта производного класса, непосредственно перед данными, описанными в самом производном классе, и сохраняют свою первоначальную последовательность (табл. 4).

Таким образом, размер объекта производного класса определяется суммарным размером данных-членов базового и этого производного класса. В случае замещения данных базового класса в объект производного класса включаются как замещенная, так и замещающая версия этих данных, что отражается на его размере и структуре выделенной для него памяти.

5. Многоуровневое наследование приводит к копированию всех открытых, закрытых и защищенных данных из базовых классов в производный класс. При этом наследовании копирование данных производится последовательно, начиная от базового класса до производного через промежуточные, добавляя к данным-членам каждого предыдущего класса данные последующего (табл. 5).

Как показали эксперименты, при замещении данных-членов одного из классов, стоящих выше в иерархии наследования, в текущий и во все нижестоящие классы копируется как замещенная, так и замещающая копия

данного. В результате размер объекта класса, находящегося в конце цепочки наследования, равен сумме размеров всех данных-членов объявленных в каждом из вышестоящих классов, включая замещающие данные. Расположение данных-членов в памяти, выделенной для объекта производного класса, производится в направлении увеличения адресов памяти в порядке от базового класса вниз по иерархии до данного производного класса. Поэтому, данные, последнего производного класса оказываются наиболее удаленными от начала области памяти, выделенной для хранения его объекта.

Таблица 4

Распределение памяти для данных-членов объектов при простом наследовании

Данные классов		Объект класса cChild : cParent		Дамп памяти	
cParent	cChild	Адрес	Размер	Адрес	Контент

public:
int pint1=1;
protected:
int pint2=2;
private:
int pint3=3;
	public:
	int pint1=8;
	int int1=4;
	protected:
	int int2=5;
	private:
	int int3=6;
		0012FF64	28 байт	0012FF64
				0012FF68
				0012FF6C
				0012FF70
				0012FF74
				0012FF78
				0012FF7C
					00000001
					00000002
					00000003
					00000008
					00000004
					00000005
					00000006

Таблица 5

Размещение в памяти данных объектов при многоуровневом наследовании

Данные классов			Наследование	Объект класса cChild	
cBase	cParent	cChild		Адрес	Размер

public:
int int1=1;
protected:
int int2=2;
private:
int int3=3;
	public:
	int bint2=10;
	int pint1=4;
	protected:
	int pint2=5;
	private:
	int pint3=6;
		public:
		int pint2=11;
		int int1=7;
		protected:
		int int2=8;
		private:
		int int3=9;
			class cParent :
			public cBase
			
			class cChild :
			public cParent
				0012FF54	44 байта

Дамп памяти					
Адрес	Контент	Адрес	Контент	Адрес	Контент
0012FF54 0012FF58 0012FF5C 0012FF60	00000001 00000002 00000003 0000000a	0012FF64 0012FF68 0012FF6C 0012FF70	00000004 00000005 00000006 0000000b	0012FF74 0012FF78 0012FF7C	00000007 00000008 00000009

6. Множественное наследование классов (открытое и закрытое) приводит к копированию в производный класс данных-членов всех базовых классов. При этом порядок их копирования и расположения в памяти, выделенной для объекта производного класса, определяется порядком описания наследования их классов, определенном при объявлении производного класса. Вопросы определения размеров объектов производных

классов и реализация в памяти замещаемых в производных классах данных-членов при множественном наследовании идентичны соответствующим вопросам при многоуровневом наследовании (табл. 6).

Таблица 6

Размещение в памяти данных объектов при множественном наследовании

Данные классов			Объект класса cChild		
cParent1	cParent2	cChild	Наследование	Адрес	Размер

public:
int p1int1=1;
protected:
int p1int2=2;
private:
int p1int3=3;
	public:
	int p2int1=4;
	protected:
	int p2int2=5;
	private:
	int p2int3=6;
		public:
		int p2int2=10;
		int int1=7;
		protected:
		int int2=8;
		private:
		int int3=9;
			class cChild :
			public
			cParent1,
			public
			cParent2
				0012FF58	40 байт

Дамп памяти					
Адрес	Контент	Адрес	Контент	Адрес	Контент
0012FF58 0012FF5C 0012FF60 0012FF64	00000001 00000002 00000003 00000004	0012FF68 0012FF6C 0012FF70 0012FF74	00000005 00000006 0000000a 00000007	0012FF78 0012FF7C	00000008 00000009

При многоуровневом множественном наследовании правила выделения и распределения памяти определяются правилами для многоуровневого и множественного наследования, что отражается как на размере объектов, так и на структуре выделяемой для них памяти.

Выводы. В статье рассмотрены и систематизированы вопросы выделения и распределения памяти для объектов простых классов для статических и нестатических свойств, а также распределение памяти для данных-членов при построении основных видов наследования.

Список литературы: 1. *Уильям Топп, Уильям Форд.* Структуры данных в C++. – М.: ЗАО Издательство БИНОМ, 1999. – 816 с. **2.** *Фурман І.О., Краснобаєв В.А., Далека В.Д.* Моделі та структури даних у системах автоматизованого керування: Підручник для ВНЗ. – К., 2004. – 253 с. **3.** *Amjad Z.* ATL Under the Hood Part 1 http://www.codeguru.com/cpp/comtech/atl/tutorials/article.php/c3607/- 04/03/2002. 4. A*mjad Z.* ATL Under the Hood Part 2 http://www.codeguru.com/cpp/comtech/atl/tutorials/article.php/c3545/- 02/15/2002. **5.** *Amjad Z.* ATL Under the Hood Part 3 http://www.codeguru.com/cpp/comtech/atl/tutorials/article.php/c3549/- 03/29/2002. **6.** *Amjad Z.* ATL Under the Hood Part 5. http://www.codeguru.com/cpp/comtech/atl/tutorials/article.php/c3547/-10/18/2002**. 7.** *Шилд Герберт***.** Полный справочник по C++. – М.: Издательский дом Вильямс, 2004. – 800 с. **8.** *Гук М.* Аппаратные средства IBM PC. Энциклопедия. – СПб.: Питер, 2006. – 1072 с.

Поступила в редакцию 20.04.2007